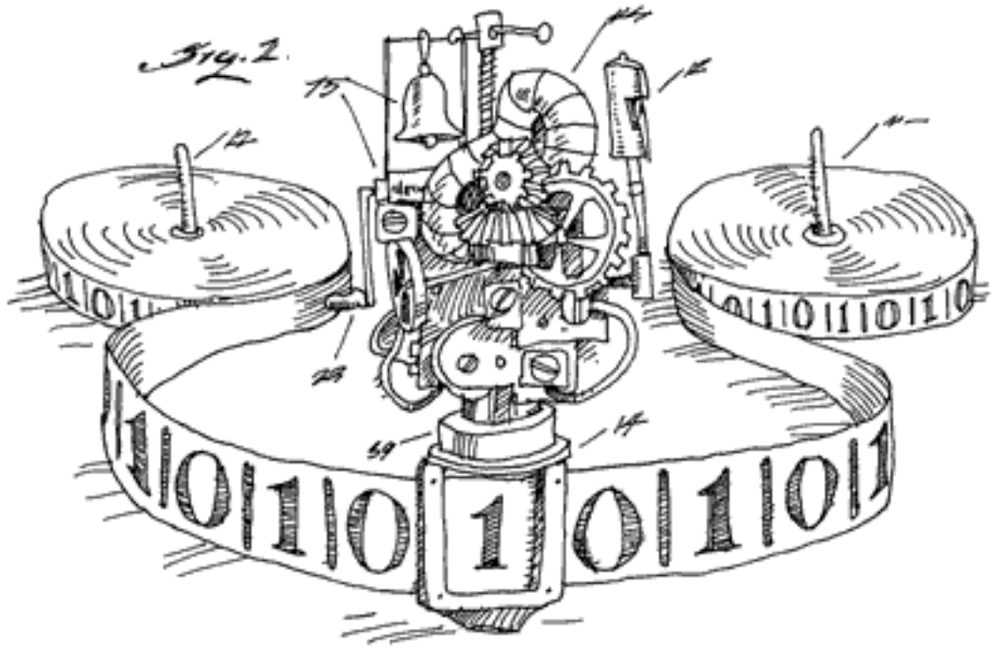
CSC4010: Introduction to Theoretical Computer Science



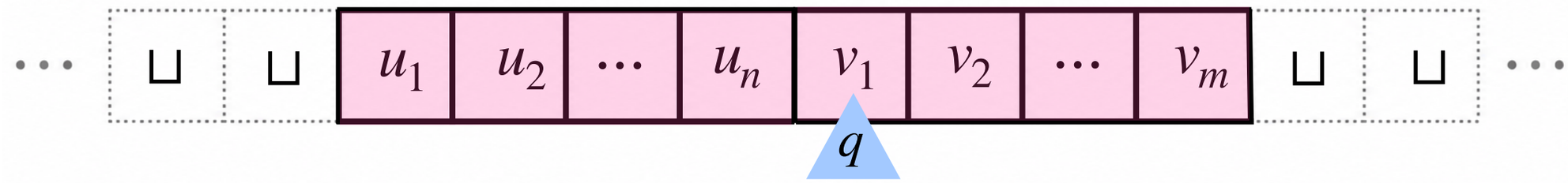
Lecture 5

Administrative Things

- Quiz in Tutorial on 29/9/2026.
- First 20 minutes of class.
- 16 points
- True/False, MCQ, short answer type questions.
- Based on lectures 1-6

Recap

- Turing Machine is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$.
- A configuration uqv (where $u = u_1u_2\cdots u_n \in \Sigma^*$, $v = v_1v_2\cdots v_m \in \Sigma^*$ and $q \in Q$) is the current status of the Turing Machine.



- Given C_i, $C_{i+1} = \text{NEXT}(C_i)$ is determined by δ.
- Computational history of a Turing machine on an input $w \in \{0,1\}^*$ is a series of configurations $C_1, C_2, \cdots$ where $C_{i+1} = \text{NEXT}(C_i)$.
- Given a decidable language L, the language $\bar{L} = \{x \mid x \in \{0,1\}^* \backslash L\}$ is also decidable.

Church-Turing Thesis

Let $Y \subseteq \{0,1\}^*$ be a language.

Church-Turing Thesis: There exists an "algorithm" or a "procedure" for figuring out whether a $x \in Y$ if and only if there is a Turing Machine that decides Y.

Alonzo Church

Alan Turing

Church-Turing Thesis: What it says?

- The Church-Turing Thesis says:
  - The Turing Machine model is the "correct" way of modeling arbitrary computation.
  - The informal concept of an "algorithm" is successfully modeled by the rigorous definition of a Turing Machine.

Church-Turing Thesis: What it says?

- The Church-Turing Thesis says:
  - The Turing Machine model is the "correct" way of modeling arbitrary computation.
  - The informal concept of an "algorithm" is successfully modeled by the rigorous definition of a Turing Machine.

This is not a theorem - it is a falsifiable scientific hypothesis. And it has been thoroughly tested!

Can we verify the Church-Turing Thesis?

Left-Right-Stationary Turing Machine

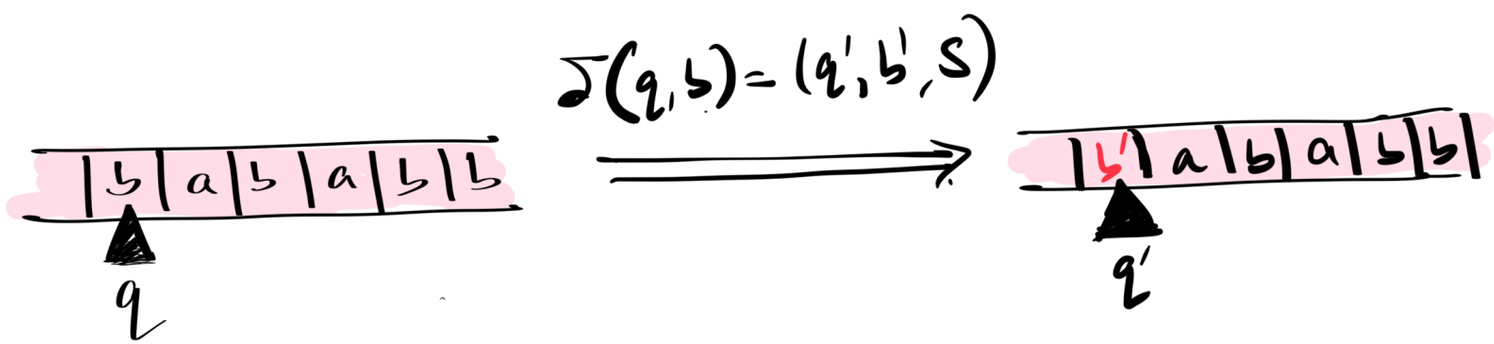
Mathematically, a Left-Right-Stationary Turing Machine is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ such that:

- Q is a finite set (the set of "states".)
- Σ is a finite set of symbols ("the tape alphabet").
- δ is a function $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R, S\}$. (The "transition function".)
- S means the head will not move in this step.

Left-Right-Stationary Turing Machine

Mathematically, a Left-Right-Stationary Turing Machine is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ such that:

- Q is a finite set (the set of "states".)
- Σ is a finite set of symbols ("the tape alphabet").
- δ is a function $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R, S\}$. (The "transition function".)
- S means the head will not move in this step.



Left-Right-Stationary TM

- The model is still realistic, even though we added an extra feature.
- Is it a counterexample to the Church-Turing thesis?
- No!
- Let’s prove the Left-Right-Stationery Turing Machine Model is equivalent to the original Turing Machine model.

Left-Right-Stationary Turing Machine

- Let $L \subseteq \{0,1\}^*$ be a language.

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Left-Right-Stationary Turing Machine

- Let $L \subseteq \{0,1\}^*$ be a language.

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Leftarrow$) M'_L is an L-R-S Turing Machine that does not use the "stationary function".

Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Left-Right-Stationery Turing Machine

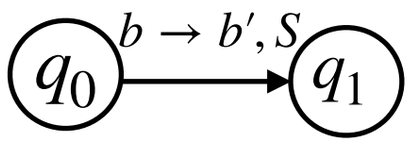
Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.

Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

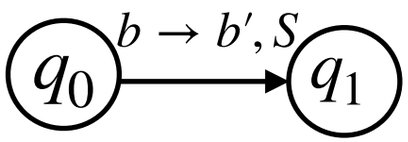
Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.

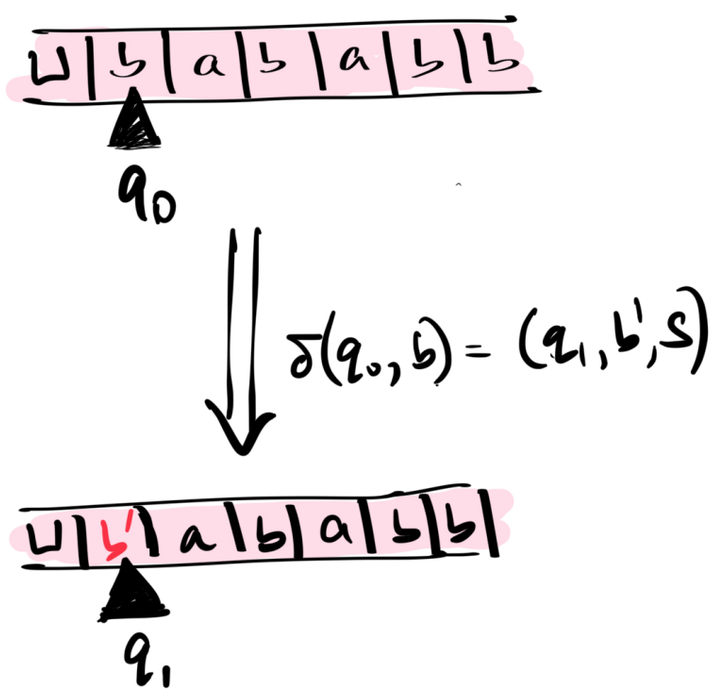


Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.

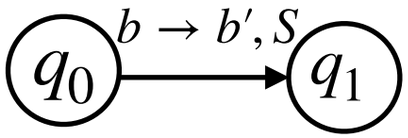


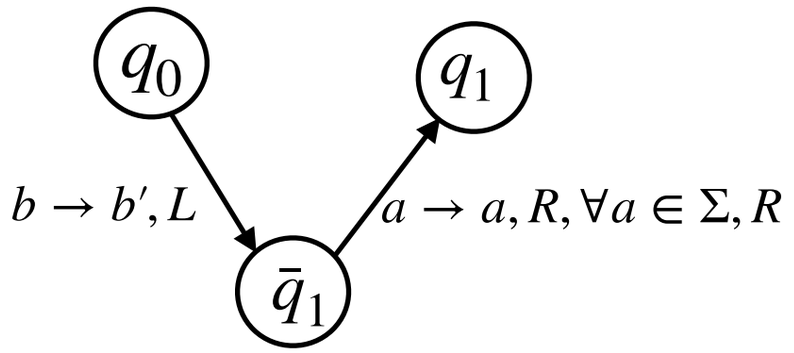


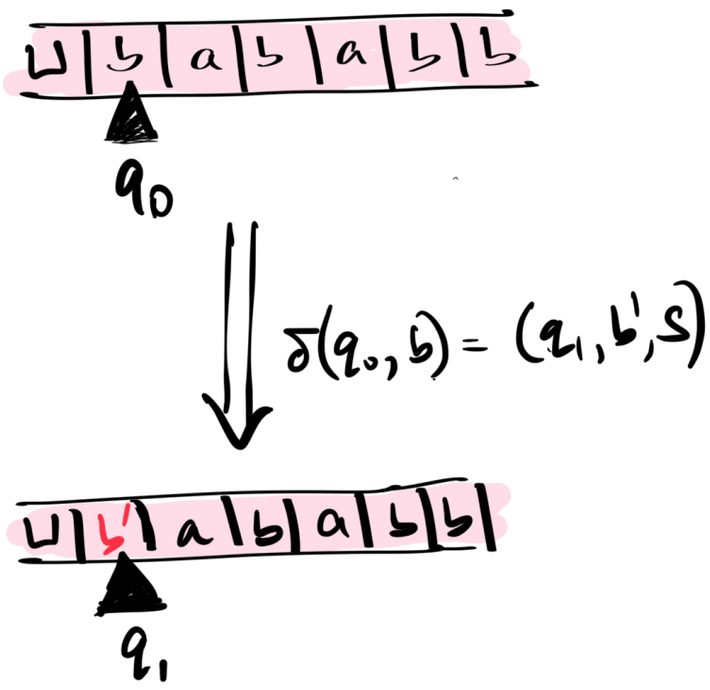
Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.



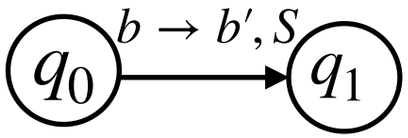


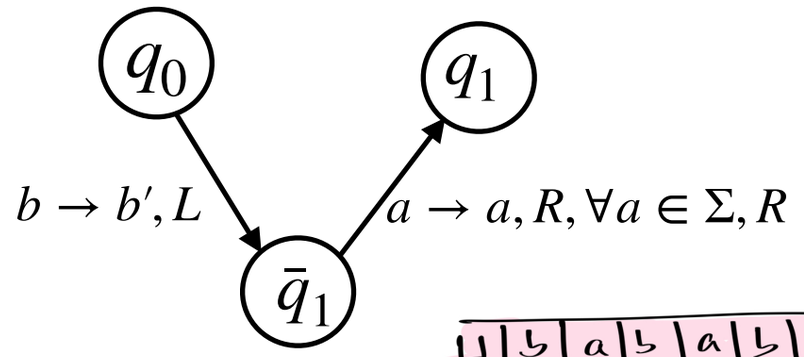


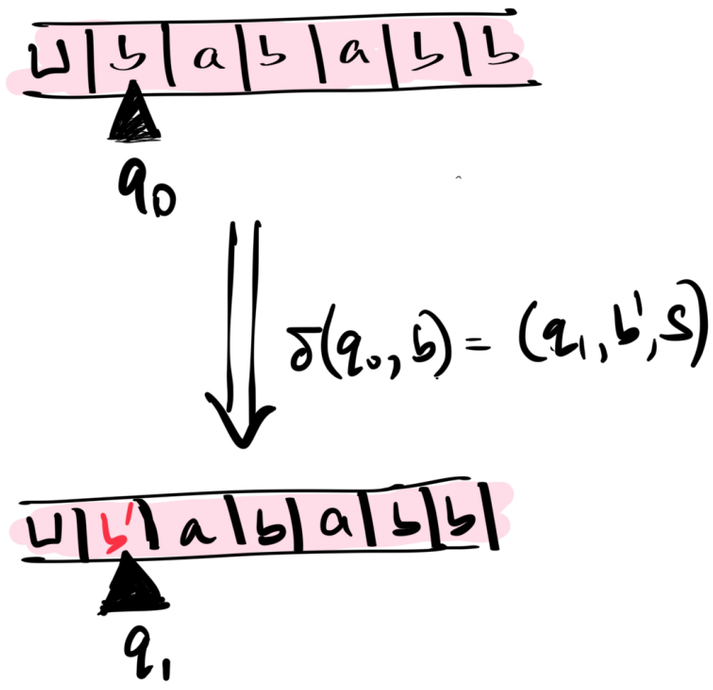
Left-Right-Stationery Turing Machine

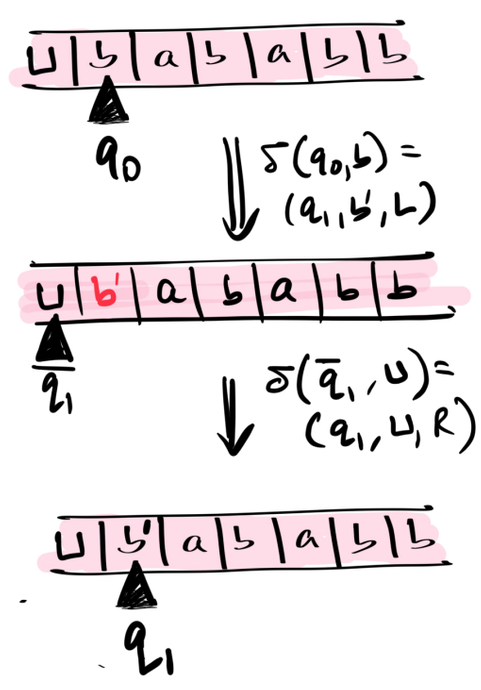
Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Rightarrow$) A proof idea: Simulate S by L followed by an R.









Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Rightarrow$) Suppose $M_L = (Q, \Sigma, \delta, q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$ is an L-R-S Turing Machine deciding L. Then, $M'_L = (Q', \Sigma, \delta', q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$?

Left-Right-Stationery Turing Machine

Theorem: There exists a L-R-S Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Rightarrow$) Suppose $M_L = (Q, \Sigma, \delta, q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$ is an L-R-S Turing Machine deciding L. Then, $M'_L = (Q', \Sigma, \delta', q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$?

Check rigorously that M'_L decides L.

Turing Machine with Reset

Turing Machine with Reset

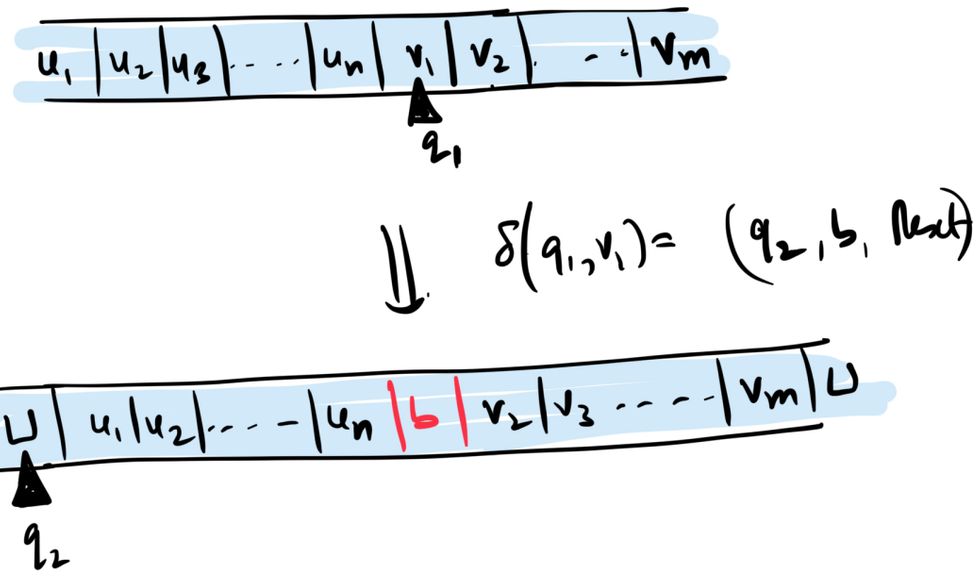
Mathematically, a Turing Machine with Reset is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ such that:

- Q is a finite set (the set of "states".)
- Σ is a finite set of symbols ("the tape alphabet").
- δ is a function $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R, \text{Reset}\}$. (The "transition function".)
- This implies: if $C_i = uq_1v$ and $\delta(q_1, v_1) = (b, q_2, \text{Reset})$, then $\text{NEXT}(C_i) = q_2 \sqcup ubv_2v_3 \cdots v_m$.

Turing Machine with Reset

Mathematically, a Turing Machine with Reset is a 7-tuple $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ such that:

- Q is a finite set (the set of "states".)
- Σ is a finite set of symbols ("the tape alphabet").
- δ is a function $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R, \text{Reset}\}$. (The "transition function".)
- This implies: if $C_i = uq_1v$ and $\delta(q_1, v_1) = (b, q_2, \text{Reset})$, then $\text{NEXT}(C_i) = q_2 \sqcup ubv_2v_3 \cdots v_m$.



Turing Machine with Reset

Let $L \subseteq \{0,1\}^*$ be a language.

Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Turing Machine with Reset

Let $L \subseteq \{0,1\}^*$ be a language.

Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Leftarrow$) M'_L is a Turing Machine with Reset that does not use the "Reset function".

Turing Machine with Reset

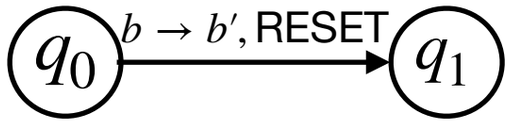
Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.

Turing Machine with Reset

Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

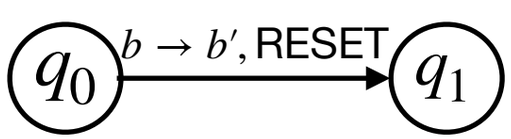
Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.

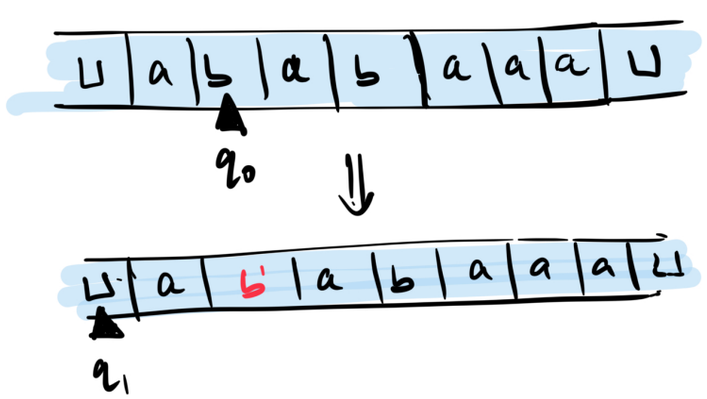


Turing Machine with Reset

Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.

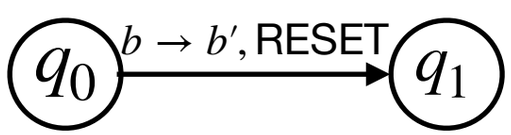


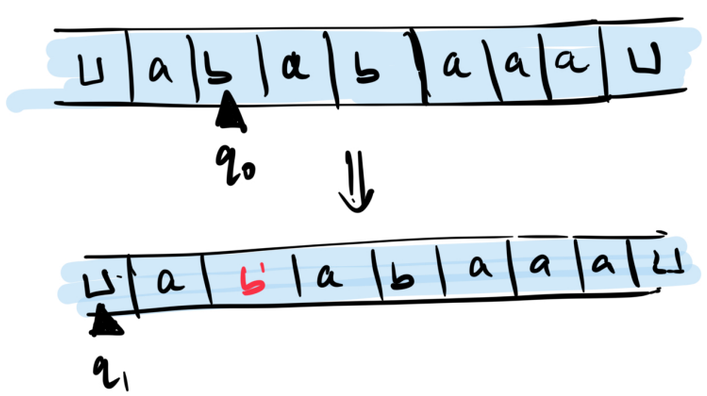


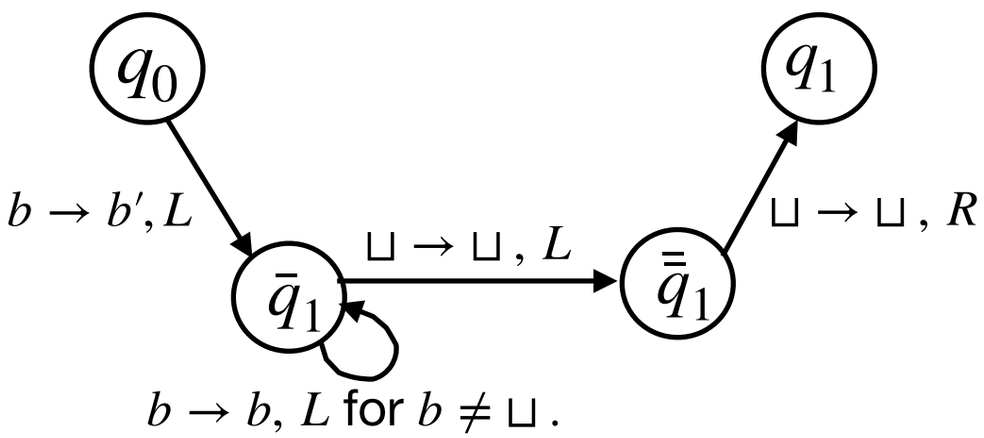
Turing Machine with Reset

Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.



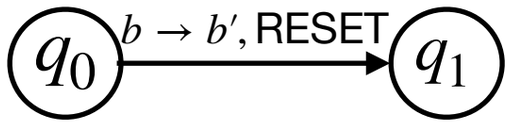


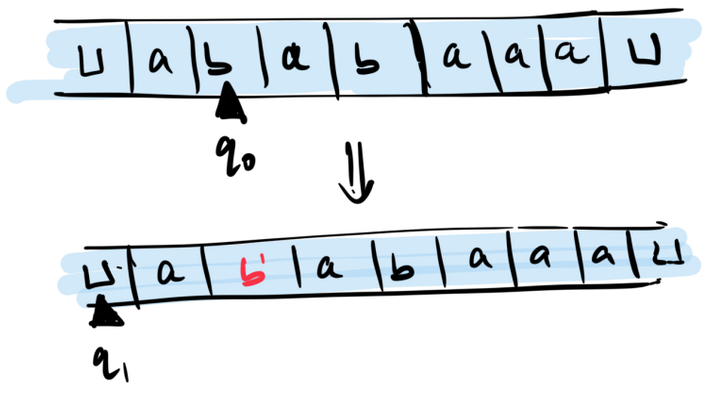


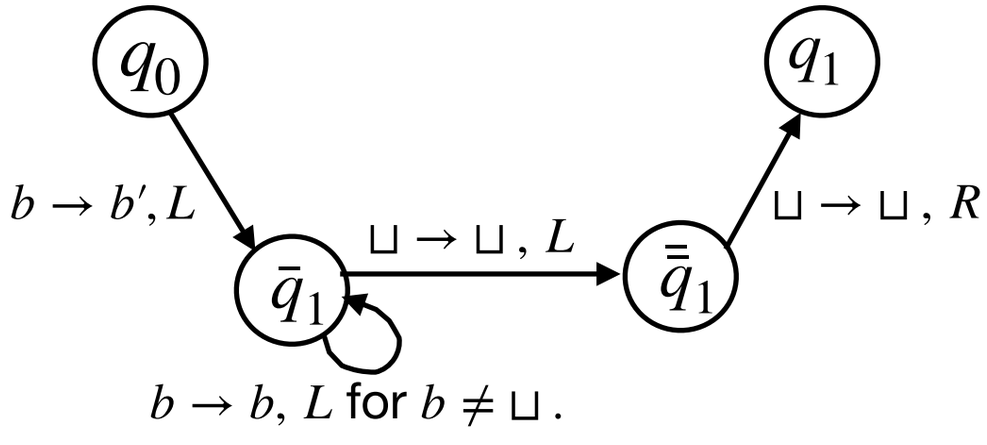
Turing Machine with Reset

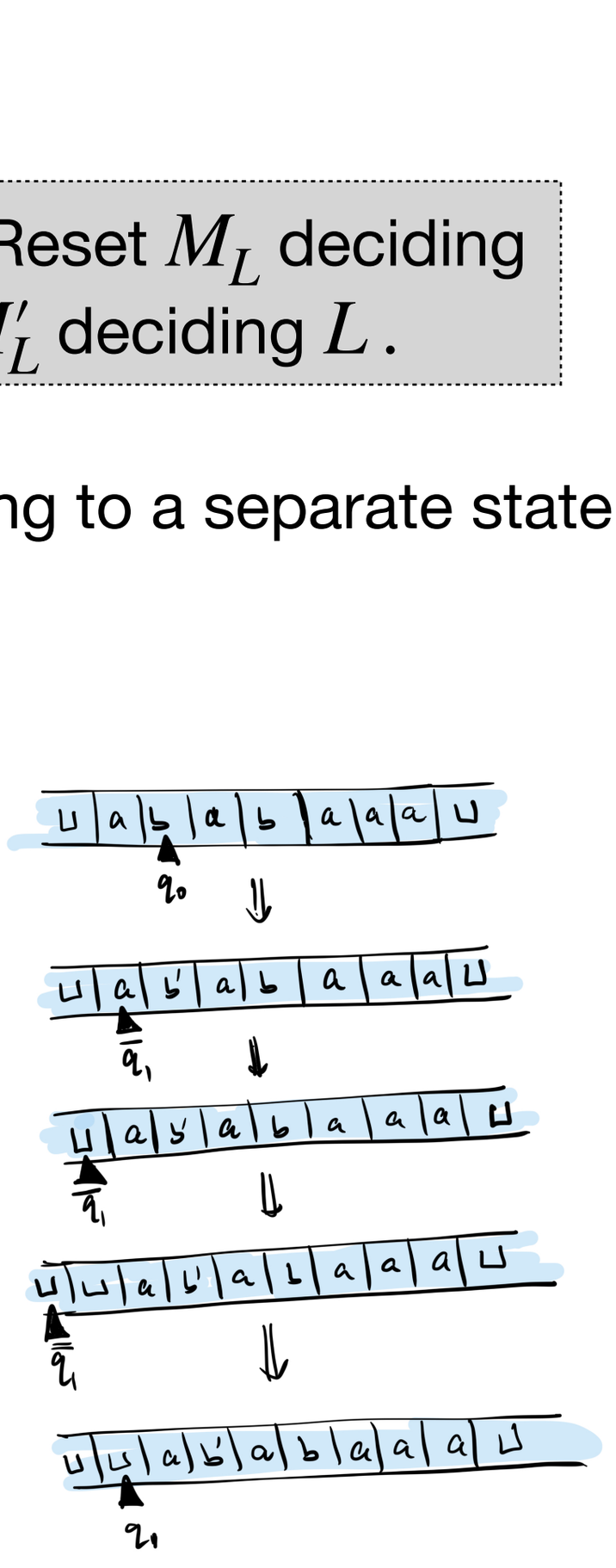
Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Leftarrow$) A proof idea: Simulate RESET by going to a separate state and rewinding.









Turing Machine with Reset

Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Rightarrow$) Suppose $M_L = (Q, \Sigma, \delta, q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$ is a Turing Machine with RESET deciding L. Then, $M'_L = (Q', \Sigma, \delta', q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$?

Turing Machine with Reset

Theorem: There exists a Turing Machine with Reset M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

Proof: ($\Rightarrow$) Suppose $M_L = (Q, \Sigma, \delta, q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$ is a Turing Machine with RESET deciding L. Then, $M'_L = (Q', \Sigma, \delta', q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$?

Check rigorously that M'_L decides L.

Multi-Tape Turing Machine

2-Tape Turing Machine

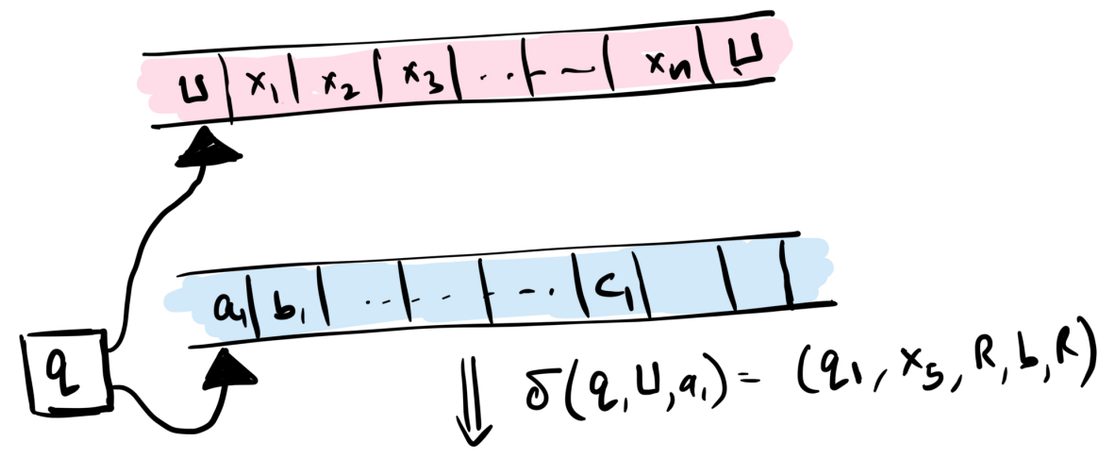
- A two-tape Turing Machine is like an ordinary Turing Machine with two tapes.

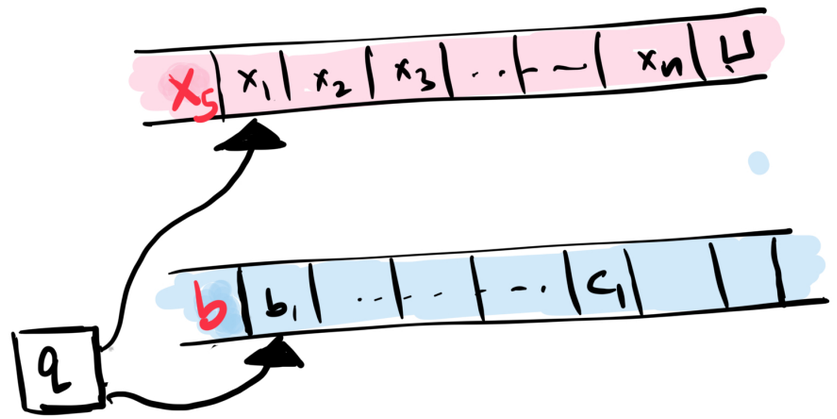
- Each tape has its own head for reading and writing.

- Initially, the input appears on one tape, and the second tape is blank.

- The transition function allows for reading, writing, and moving the head on all the tapes simultaneously:

$$\delta : Q \times \Sigma^2 \rightarrow Q \times \Sigma \times \{L, R\} \times \Sigma \times \{L, R\}$$





2-Tape Turing Machine

Let $L \subseteq \{0,1\}^*$ be a language.

Theorem: There exists a 2 Tape Turing Machine M_L deciding L if and only if there exists a Turing Machine M'_L deciding L.

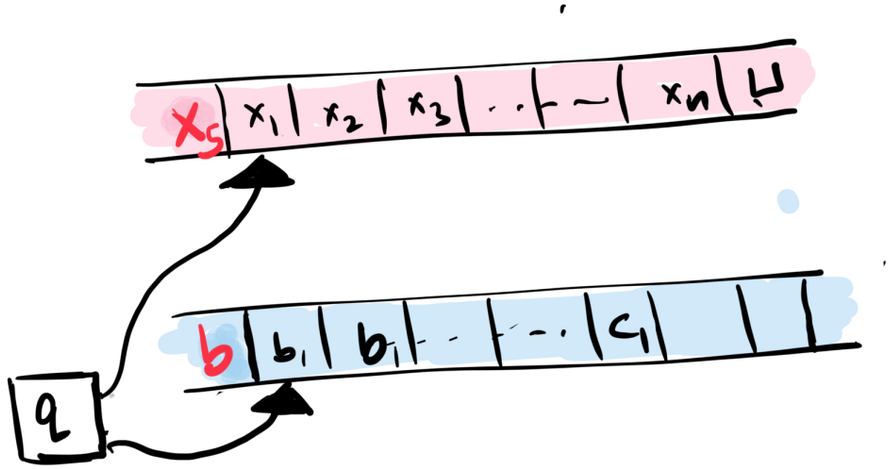
Proof: ($\Leftarrow$) M'_L is a 2 Tape Turing Machine that does not use the second tape.

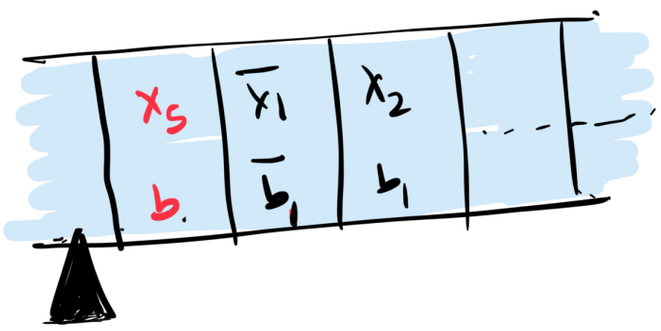
2 Tape Turing Machine

Proof: ($\Rightarrow$) Suppose $M_L = (Q, \Sigma, \delta, q_{\text{acc}}, q_{\text{rej}}, q_0, \sqcup)$ is a 2 Tape Turing Machine deciding L. Then, $M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Idea:

- Divide the single tape into two tracks by adding two symbols in a single cell. Achieved by expanding the alphabet.
- To simulate two heads, use different symbols to denote the presence of "virtual heads".
- The real head goes back and forth, updating the position of the virtual heads.





$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

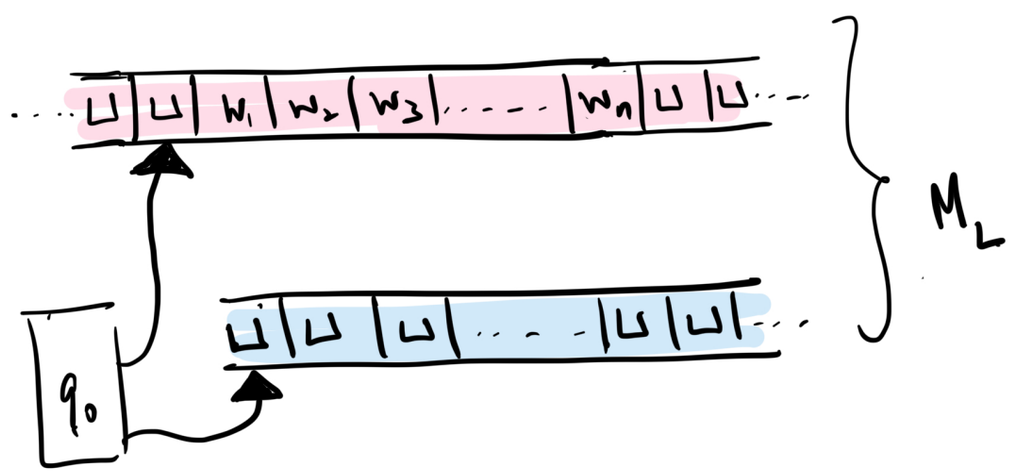
Simulating 2-Tape TM: Tape Alphabet

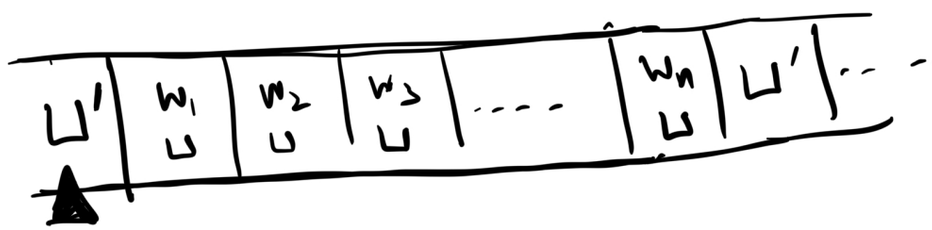
- To indicate presence of simulated head, define $\underline{\Sigma} = \{\underline{b} \mid b \in \Sigma\}$. A letter $\underline{b}$ denotes the presence of virtual head on the symbol.

- New blank symbol $\sqcup'$.

- New alphabet $\Sigma' = \{ \sqcup' \} \cup (\Sigma \cup \underline{\Sigma})^2$.

- For example: If the input to M_L is $w_1 w_2 \cdots w_n$. Then, the input to M'_L is $(w_1, \sqcup), (w_2, \sqcup), \cdots, (w_n, \sqcup)$.





$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the underlined symbols and remember these symbols that are underlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)

$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

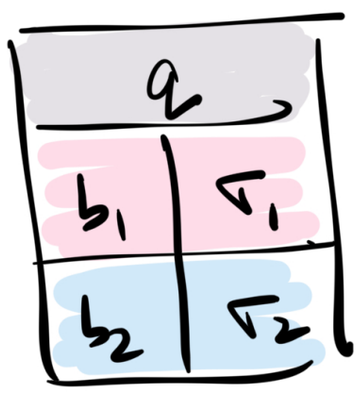
Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the underlined symbols and remember these symbols that are underlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)

An Example of a state in M'

$q' =$



$q \in Q$

$b_1, b_2 \in \Sigma \cup \{?\}$

$\sigma_1, \sigma_2 \in (\Sigma \times \{L, R, S\}) \cup \{?\}$

$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the underlined symbols and remember these symbols that are underlined.

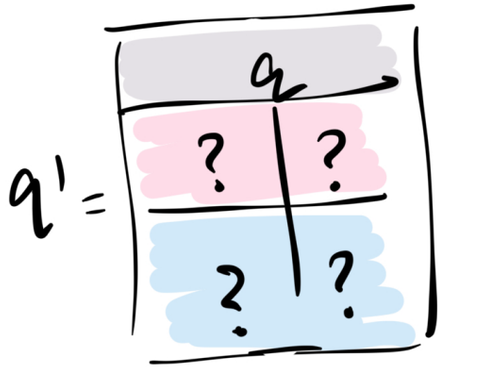
- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)

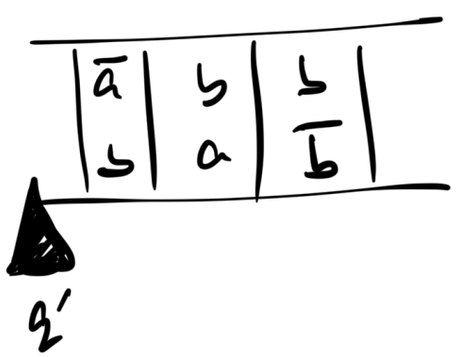
$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the underlined symbols and remember these symbols that are underlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)



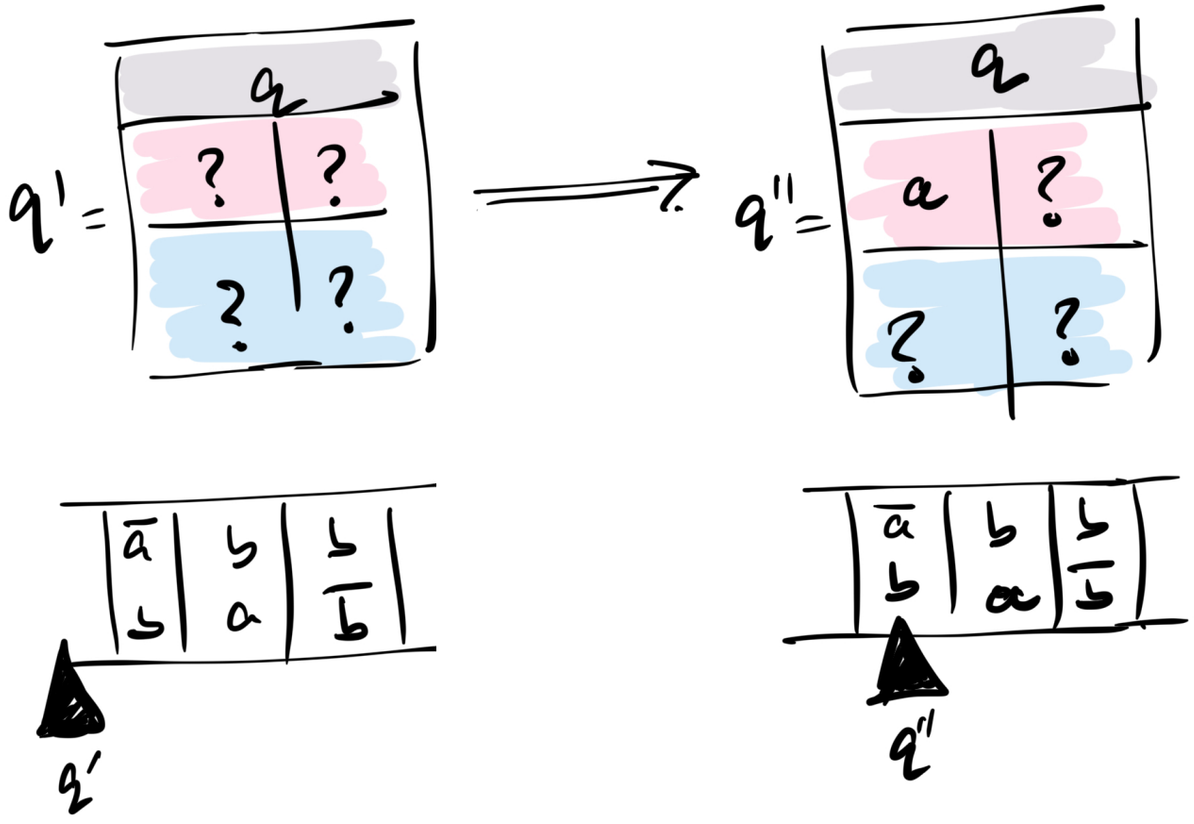


$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the underlined symbols and remember these symbols that are underlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)

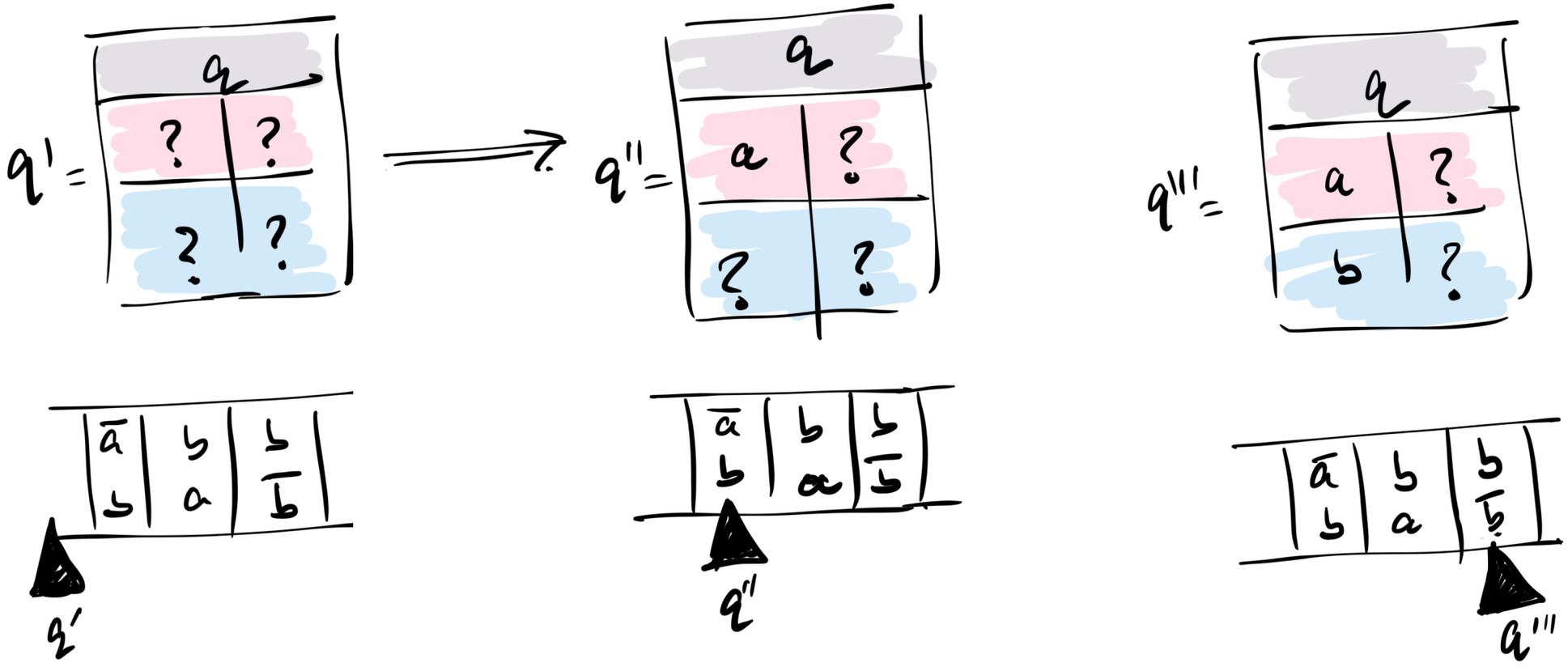


$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the underlined symbols and remember these symbols that are underlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)

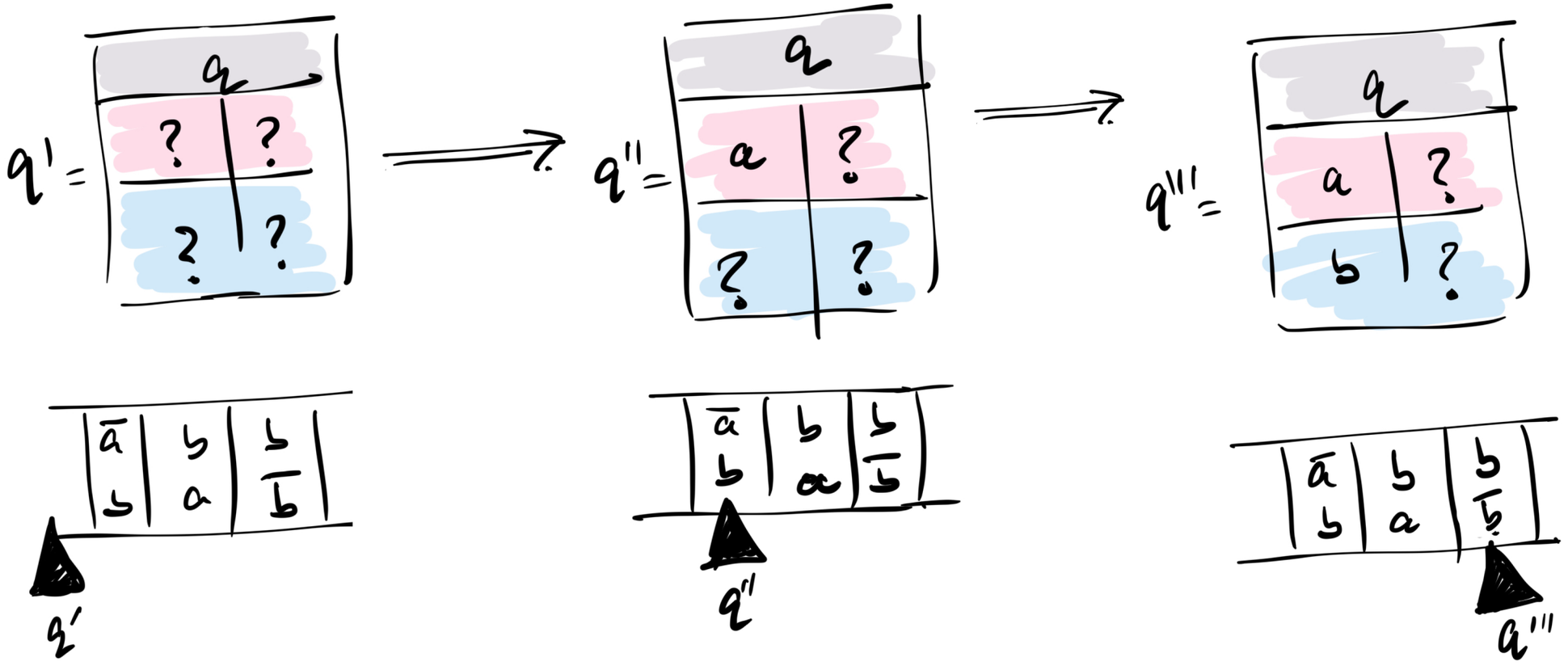


$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

Simulating 2-Tape TM: Transition and States

- Scan the tape using the real head to find the underlined symbols and remember these symbols that are underlined.

- (Because there are only a finite number of possibilities, the machine can 'remember' these symbols via a finite number of states, without writing anything to the tape.)



$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M .

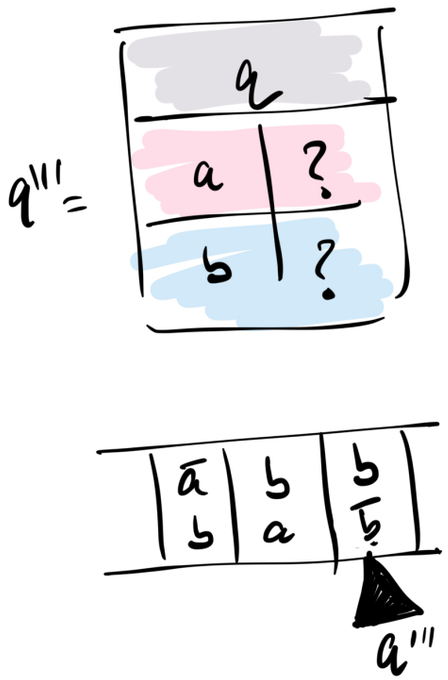
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.



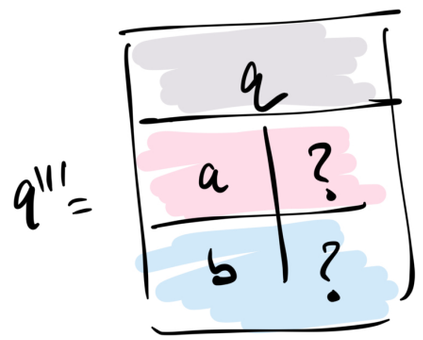
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

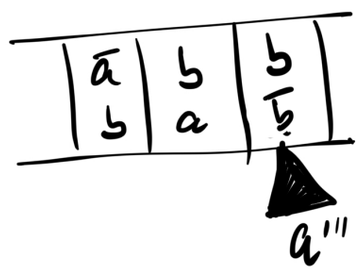
Simulating 2-Tape TM: Transition and States

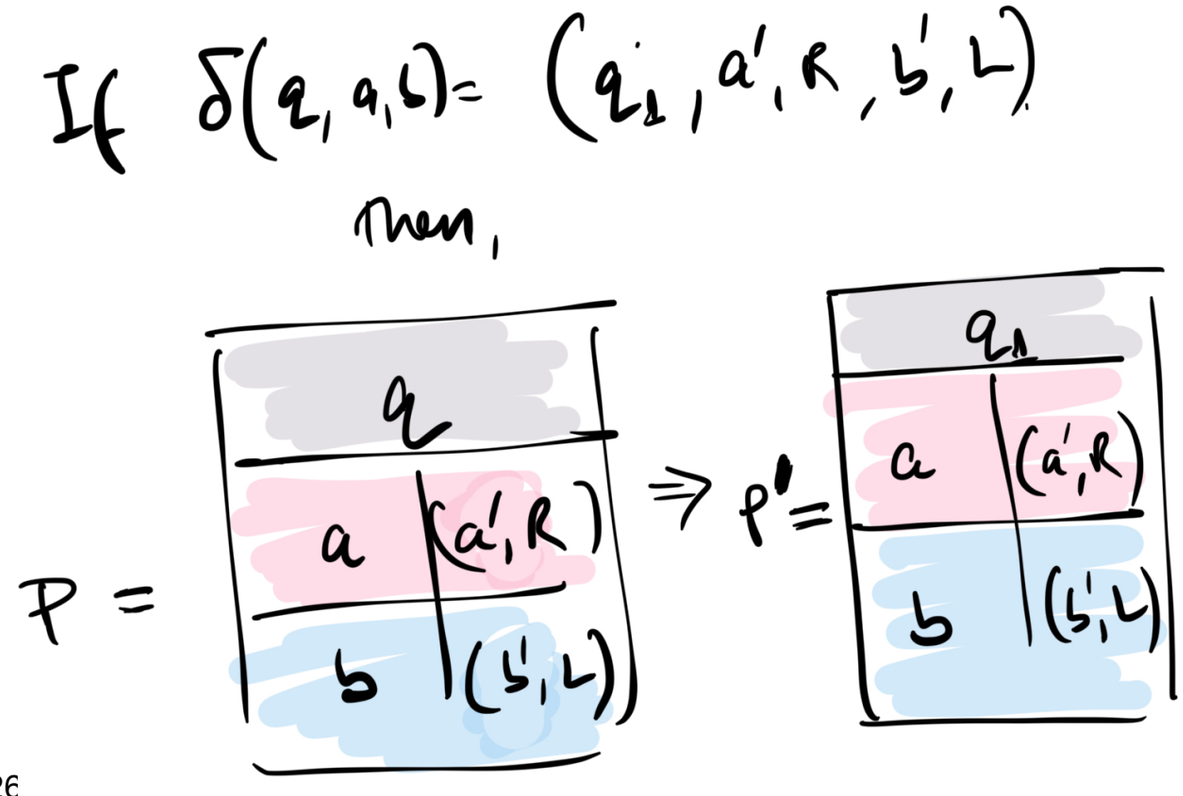
- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.







$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

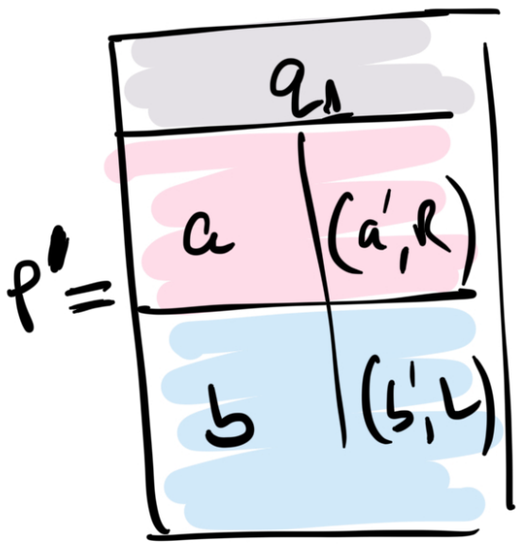
- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.

$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).



- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.

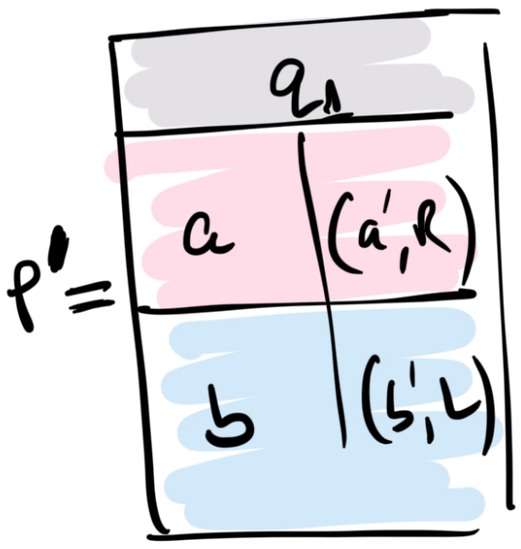
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

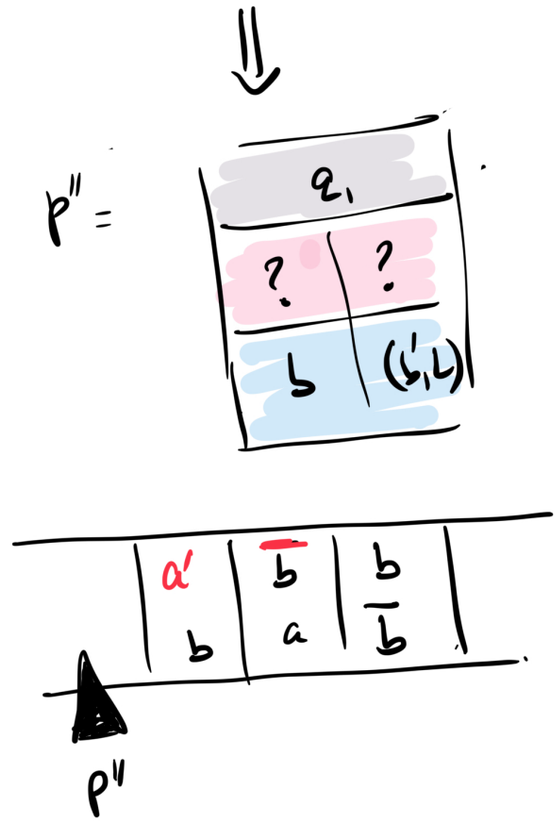
Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.





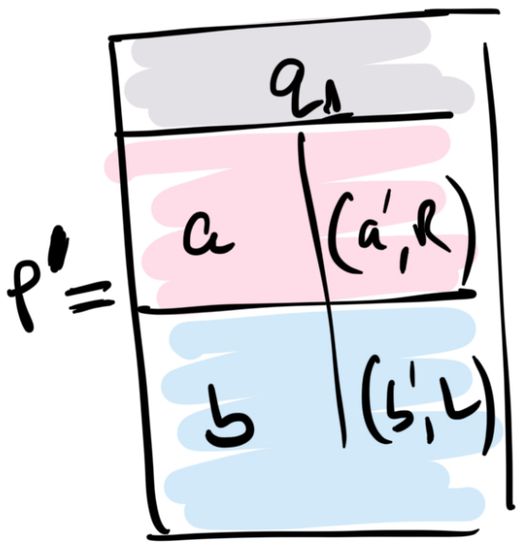
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

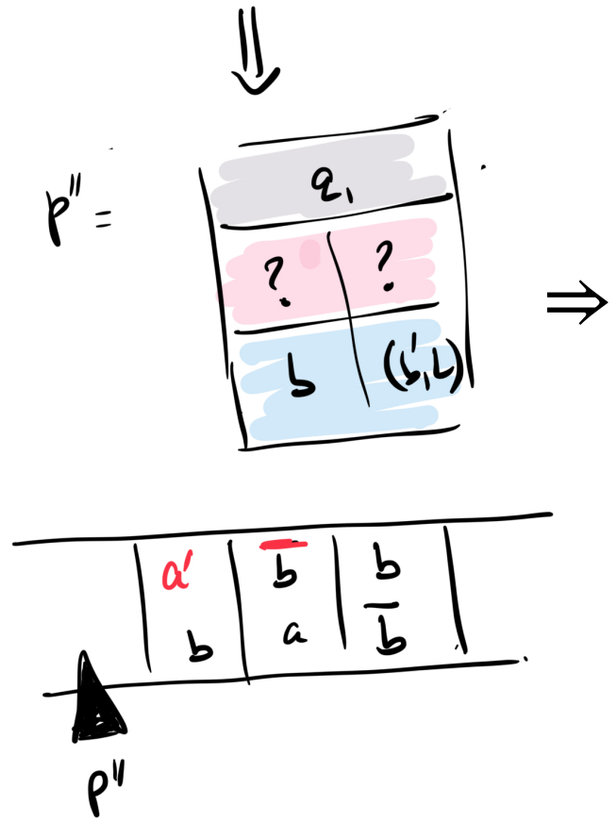
Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

- Move the virtual heads to the right locations as specified by the transition function of M.





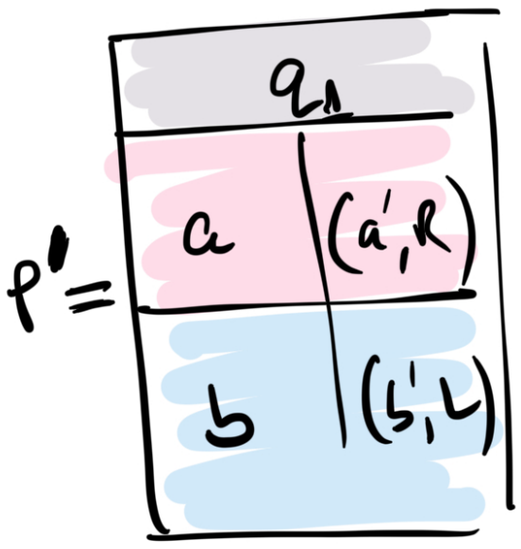
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

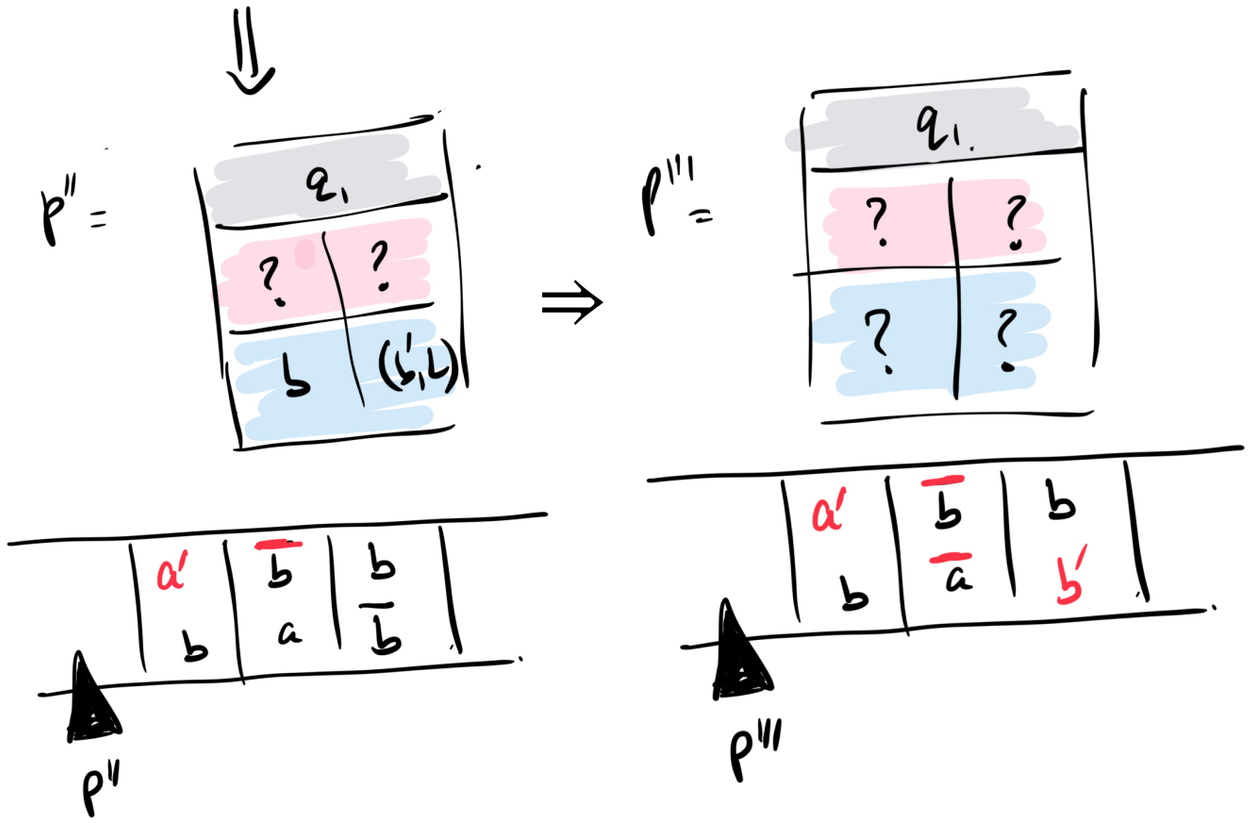
Simulating 2-Tape TM: Transition and States

- Transition to a new state according to δ of the two-tape machine (depending on the current state and the symbols under the virtual head).

- Replace the symbols under the virtual head with the symbols to be written to each tape, as specified by δ.

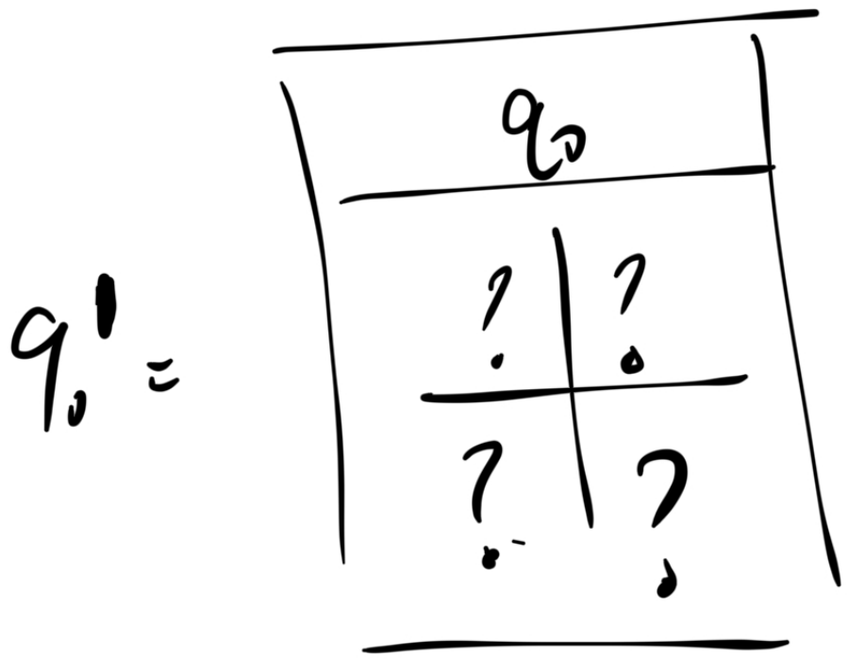
- Move the virtual heads to the right locations as specified by the transition function of M.





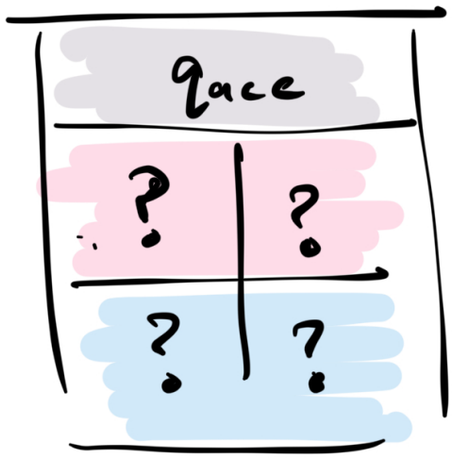
$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')$?

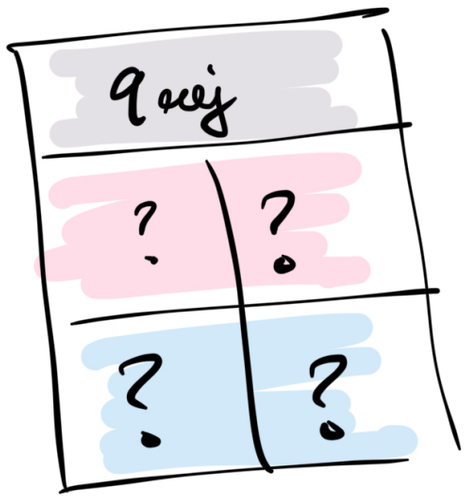
Simulating 2-Tape TM: Initial State

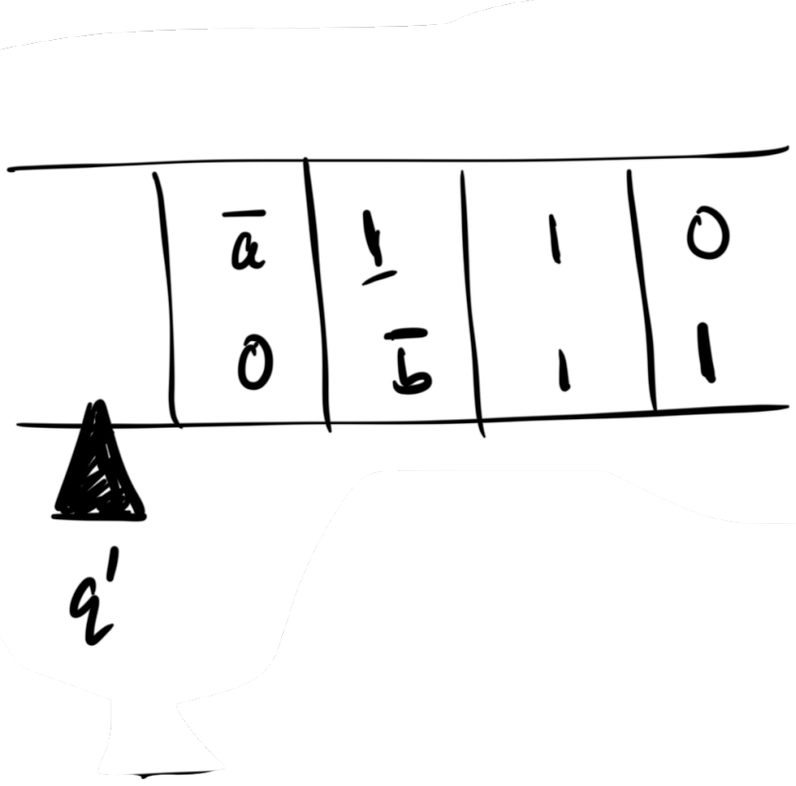


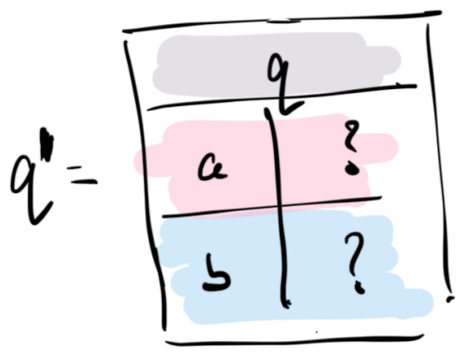
$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$

Simulating 2-Tape TM: Halting States





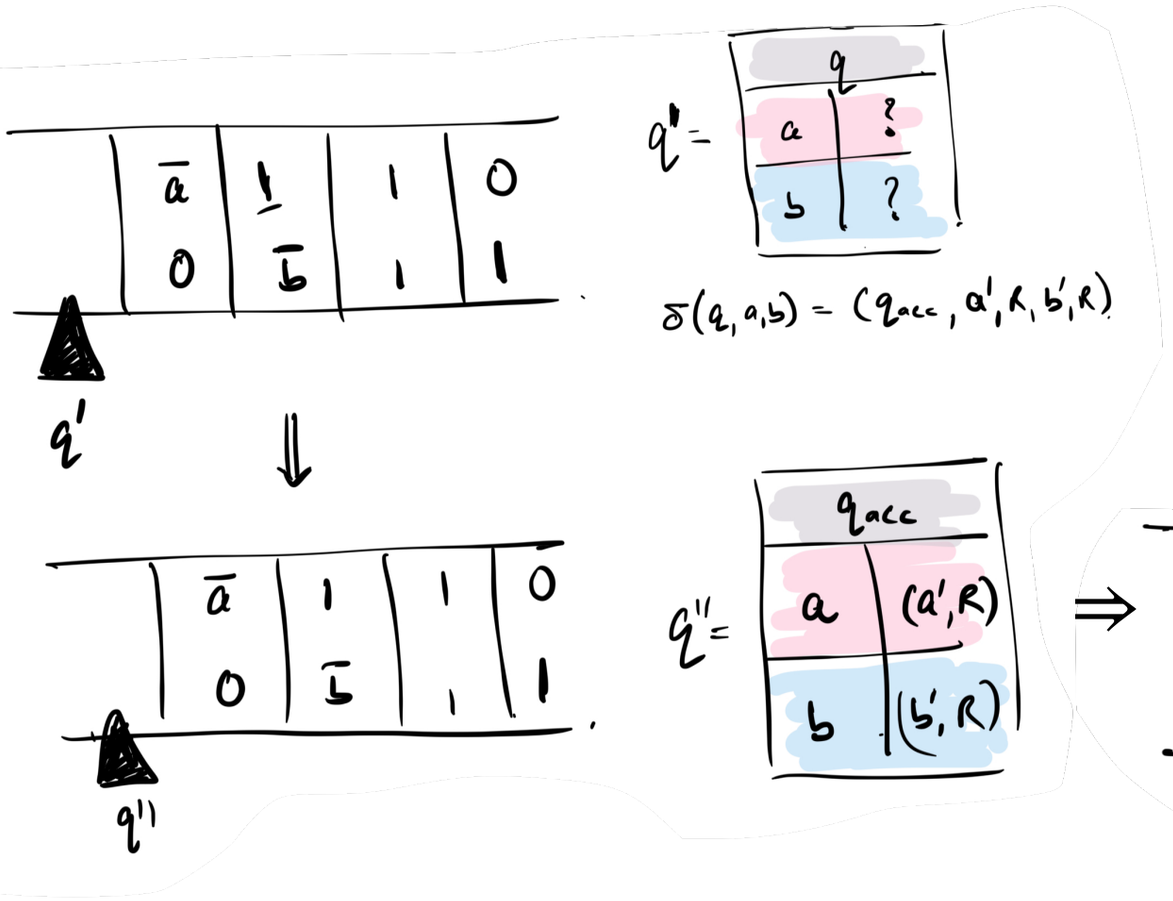




$\delta(q, a, b) = (q_{acc}, a', R, b', R)$

Simulating 2-Tape TM: Halting States

$$M'_L = (Q', \Sigma', \delta', q'_{\text{acc}}, q'_{\text{rej}}, q'_0, \sqcup')?$$



Some Thoughts on the Proof Sketch

- Constructing this one-tape machine from the definition of the two-tape machine is a tedious, but completely mechanical, process.

- The important point is that the one-tape machine perfectly simulates the operation of the original two-tape machine, even though it typically takes many steps to carry out what the original machine does in a single step.

- If the two-tape machine accepts an input, then the one-tape simulation will do so as well, and similarly if the original machine rejects or loops.

Summary: TMs can simulate all “reasonable” machines

We can add other “features” to the basic Turing Machine model:

- The ability to observe two neighboring cells.
- **Multiple-head TM:** there are multiple heads operating independently on a single tape. Here,
  $\delta : Q \times \Sigma \times \Sigma \to Q \times \Sigma \times \{L, R\} \times \Sigma \times \{L, R\}$.

None of these changes have any affect on the model.